

Pipeline:

Uno dei problemi principali della pipeline è assicurare un flusso continuo di istruzione alla pipeline, trattando in maniera ottimale i salti condizionati.

Per trattare i salti condizionati sono stati considerati vari modi:

flussi multipli: vengono eseguite nella pipeline entrambe le istruzioni generando 2 flussi, questo può dare problemi di contesa delle risorse e di ulteriori salti condizionati in ciascun flusso.

prefetch branch target: viene sempre prelevato l'indirizzo di destinazione del branch così che se viene eseguito il salto condizionato si ha già l'indirizzo pronto.

buffer circolare: il buffer circolare è una memoria molto veloce e piccola gestita nella fase di fetch che contiene le ultime n istruzioni prelevate. Ad ogni fetch l'hardware controlla se l'istruzione si trova nel buffer. I vantaggi sono 3:

- i) anticipando il fetch il buffer conterrà alcune istruzioni successive all'indirizzo di prelievo, quindi diminuirà il tempo per accedere alle istruzioni successive se queste sono contigue.
- ii) se si verifica un salto con una destinazione poche celle più avanti rispetto all'indirizzo del branch, questa destinazione sarà già nel buffer
- iii) nei cicli e nelle iterazioni il buffer potrà contenere tutte le istruzioni di un ciclo. casi come gli if-then-else if-then sfrutteranno al massimo il loop buffer.

branch prediction: le predizioni dei salti possono essere:

branch taken cioè predizione di saltare sempre,

branch not taken cioè l'opposto,

salto in base al codice operativo cioè i salti vengono presi o no a seconda dell'istruzione.

Questi primi 3 approcci sono statici cioè non dipendono dalla storia dell'esecuzione fino al momento dell'istruzione di salto.

bit taken/not taken vengono applicati in un registro molto veloce ad ogni istruzione di salto spingendo a considerare un salto come preso o no a seconda della storia passata. Possono essere usati uno o due bit per memorizzare la storia di un salto.

tabella della storia dei salti che tiene anche questa conto dell'esecuzione di tutti i salti.

delayed branch: vengono riordinate le istruzioni in modo che le istruzioni di salto avvengano in ritardo rispetto al momento in cui sono effettivamente desiderate.

dipendenza dai dati:

Soluzioni

- Introduzione di fasi non operative (nop)
- Individuazione del rischio e prelievo del dato direttamente all'uscita dell'ALU (data forwarding)
- Risoluzione a livello di compilatore
- Riordino delle istruzioni (pipeline scheduling)

dipendenza dai controlli:

Mettere in stallo il pipeline fino a quando non si è calcolato l'indirizzo della prossima istruzione

– Pessima efficienza, massima semplicità

- Individuare le istruzioni critiche per anticiparne l'esecuzione, eventualmente mediante apposita logica di controllo.

memoria cache:

La memoria centrale è costituita di 2^n parole indirizzabili raggruppate in blocchi di K parole, ci sono quindi $M = \frac{2^n}{k}$

blocchi. La cache consiste di C linee di K parole ciascuna. Siccome le linee sono radicalmente minori del numero di blocchi di memoria centrale in ogni momento solo un sottoinsieme di blocchi può trovarsi in cache, e quindi ogni linea non può essere dedicata esclusivamente ad un singolo blocco. Quindi ogni linea contiene un tag che specifica quale blocco sia memorizzato.

Servono comunque delle funzioni di mapping che stabiliscano una politica nell'assegnare i blocchi alla cache e nello stabilire quale blocco della memoria si trovi in quale linea della cache.

Vi sono 3 tecniche:

indirizzamento diretto o direct mapping: la più semplice esprimibile come $i = j \text{ mod } m$ dove i = linea cache, j = numero blocco, m = numero linee. Cioè ogni linea potrà contenere solo un definito insieme di blocchi in base all'indirizzo. Un drawback avviene quando un programma accede a parole di blocchi assegnati alla stessa linea, allora i blocchi vengono continuamente scambiati e si verifica un fenomeno detto *trashing*.

indirizzamento associativo: supera lo svantaggio di dover associare blocchi sempre alla stessa linea. Un blocco viene identificato unicamente con la sua etichetta e per verificare se un blocco è presente in cache si devono verificare in parallelo tutte le linee. Ciò rende la logica circuitale piuttosto complessa ma offre grande flessibilità.

indirizzamento set-associativo: è un compromesso tra i due sopra, in quanto si divide la cache in s set di k linee.

sussistono quindi le relazioni $m = s \cdot k$ e $i = j \bmod v$ dove m = linee dell'insieme della cache, j = blocco in memoria centrale, i numero dell'insieme della cache.

La logica di controllo considera quindi un indirizzo di memoria come costituito di 3 campi: tag, insieme, parola.

d bit di insieme specificano uno dei 2^d set, gli m bit dei campi tag e insieme specificano uno dei 2^m blocchi in memoria.

Dunque se ho memoria cache di $64Kb = 2^{16} B$, e i dati vengono trasferiti tra memoria centrale e cache in blocchi di 4B avrò quindi $2^{16}/4 = 16384 = 2^{14}$ linee di cache da 4B.

miss:

Dire quali sono i possibili tipi di miss in cui si può incorrere in una gerarchia di memoria:

- i) cache miss di inizio: quando un programma è all'inizio e quindi la cache è ancora vuota, è inevitabile questo tipo di miss.
- ii) compulsory misses: quando si accede ad un dato a cui non si era mai acceduto prima: quello sopra rientra in questo tipo di miss
- iii) miss per conflitti: quando più blocchi competono per lo stesso insieme, avviene nell'associazione diretta e non nelle set associative
- iv) miss per capacità: quando la cache non può contenere tutti i blocchi necessari, sono il tipo più comune di miss nelle set associative.

interrupt:

Gli interrupt sono previsti per migliorare l'efficienza del calcolatore. Siccome le periferiche sono parecchi ordini di grandezza più lente del processore questo può eseguire altre operazioni mentre la periferica esegue le sue operazioni. Quando la periferica ha finito o ha bisogno del processore il modulo di controllo della periferica invia una richiesta di interrupt al processore, questo finisce l'istruzione corrente, sospende l'operazione successiva ed invia un acknowledge al dispositivo che può quindi eliminare il suo interrupt pendente, si appresta a passare quindi il controllo alla ISR cioè la "interrupt service routine", prima deve salvare però la locazione della prossima istruzione del PC e lo stato del processore nella PSW (Program Status Word), quindi esegue la routine. Gli interrupt possono essere di vario genere:

- i) di programma se un programma ne necessita
- ii) di timer del processore per eseguire dei particolari interrupt a tempi prefissati
- iii) di I/O ad esempio se vi è un errore in trasferimento
- iv) di guasto hardware (guasto memoria, errore di parità etc etc)

Vi possono essere comunque degli interrupt multipli (cioè nidificati uno in un altro), questi possono essere gestiti in modi differenti:

- i) interrupt disabilitato
- ii) con priorità negli interrupt, cioè ogni interrupt ha una priorità e viene eseguito l'interrupt con priorità più alta

Per gestire gli interrupt ci vogliono delle apposite linee di interrupt che però non sono mai dedicate ad un singolo dispositivo, si dovrà quindi identificare quale abbia sollevato l'interrupt. Questo può essere fatto con 3 tecniche:

- i) interrogazione software (software poll): vengono interrogati tutti i moduli eseguendo una apposita routine di servizio
- ii) interrogazione hardware (daisy chain): cioè quando viene ricevuto l'interrupt si invia un ulteriore interrupt che si propaga attraverso il bus su tutti i moduli. Il modulo chiamante quando riceve questo interrupt invia sul bus dati una parola detta vettore, identificante il proprio indirizzo.
- iii) arbitraggio del bus: un modulo deve ottenere il controllo del bus prima di poter inviare l'interrupt.

bus:

le linee di bus possono essere classificate in tre categorie:

- i) bus di dati
- ii) bus di indirizzi
- iii) bus di controllo

Per utilizzare il bus un modulo deve ottenere l'uso del bus quindi inviare la richiesta al modulo con cui comunicare tramite i bus controllo e indirizzi, quindi trasferire i dati attraverso il bus dati.

Esistono due tipi di bus: dedicati e multiplexati. Nel caso vi siano bus condivisi serviranno dei metodi di arbitraggio del bus. Questi possono essere

- i) centralizzati: vi è un bus controller o arbiter che decide a chi assegnare il bus
- ii) distribuiti: si esegue un algoritmo di controllo d'accesso e tutti i controllori delle periferiche cooperano alla condivisione del bus. In ogni momento quindi verrà designato un controllore come master ed un altro come slave.

La temporizzazione inoltre può essere

- i) *sincrona*: cioè vi è un clock che determina gli eventi sul bus, e la frequenza è quindi fissa e determinata.
- ii) *asincrona*: cioè gli eventi si susseguono tenendo conto degli eventi precedenti. Il bus quindi potrà lavorare a frequenze diverse a seconda della velocità della periferica.

PCI è un esempio di bus sincrono con arbitraggio centralizzato.

ram:

i due tipi più comuni di memoria interna del calcolatore sono:

- i) *ram dinamica*: questo tipo di memoria tende continuamente a perdere la carica essendo costituita di condensatori. Serve quindi un continuo refresh dei dati scritti in memoria. I vantaggi per cui queste memorie vengono usate come memorie centrali sono che richiedono circuiti semplici e quindi sono poco costose nonostante abbiano un circuito aggiuntivo di refresh, inoltre sono più dense essendo costituite di condensatori e basta.
- ii) *ram statica*: in questa memoria si usano porte logiche per memorizzare i dati, quindi queste memorie mantengono la carica finchè ricevono corrente. Sono molto veloci nonostante il costo causato dalla complessità circuitale ed inoltre dalla bassa densità rispetto alla memoria dinamica, vengono perciò usate per piccole memorie di grande velocità come le cache.

moduli I/O:

Le funzioni dei moduli di I/O sono essenzialmente:

- i) rilevazione errori
- ii) comunicazione con altri dispositivi
- iii) comunicazione con il processore
- iv) buffering dei dati
- v) controllo e temporizzazione

vi sono 3 tecniche di I/O:

- i) *I/O da programma*: i dati vengono scambiati tra processore e modulo ed il processore, eseguendo un programma, ha accesso esclusivo alle periferiche. Se queste sono più lente del processore (e certo che sì..lo sono!) il processore aspetterà sprecando tempo prezioso
- ii) *I/O interrupt driven*: le operazioni sono guidate da Interrupt quindi il processore è slegato dal comunicare direttamente con le periferiche che proseguono il trasferimento fra di loro mentre il processore serve solo per trasferire dati con la memoria.
- iii) *I/O DMA*: come sopra però i moduli possono trasferire direttamente dati con la memoria.

ci sono inoltre due tipi di indirizzamento dei dispositivi:

- i) *memory mapped*: non vi è distinzione tra dispositivi e locazioni di memoria
- ii) *separato*: i dispositivi hanno indirizzi differenti dalla memoria

frammentazione interna esterna:

Quando la memoria viene allocata in blocchi di dimensione fissata, con l'espressione frammentazione interna si intende la differenza fra la memoria assegnata ad un processo e quella effettivamente richiesta da quest'ultimo.

Quando si alloca la memoria in blocchi di dimensione variabile e si caricano e si rimuovono da quest'ultima dei processi, lo spazio libero si frammenta in piccole parti e si parla di frammentazione esterna quando lo spazio libero complessivo nella memoria è sufficiente per soddisfare una richiesta, ma non è contiguo.

interruzione precisa:

Un'interruzione si dice precisa quando gode delle seguenti quattro proprietà:

- (a) il program counter viene salvato in un posto noto,
- (b) tutte le istruzioni che precedono quella puntata dal program counter sono state completamente eseguite,
- (c) nessuna delle istruzioni che seguono quella puntata dal program counter è stata eseguita,
- (d) lo stato di esecuzione dell'istruzione puntata dal program counter è noto.

Belady:

Per anomalia di Belady si intende il fenomeno per cui, nonostante si incrementi la memoria fisica disponibile e quindi il numero di frame totali, non è detto che i page fault diminuiscano. Un algoritmo di rimpiazzamento delle pagine che soffre di questo problema è l'algoritmo FIFO (First-In First-Out) che registra in una coda FIFO le pagine caricate in memoria. In questo modo in testa alla coda si troverà la pagina che da più tempo è presente in memoria; al momento della scelta della pagina da rimpiazzare si selezionerà quindi l'elemento in testa alla coda, mentre la nuova pagina caricata verrà a

inserita in fondo alla coda stessa.

microprog:

L'interpretazione del microprogramma viene usualmente eseguita da due reti sequenziali LLC interagenti, denominate Parte Controllo (PC) e Parte Operativa (PO). La PC costituisce l'automa di controllo dell'unità. Essa ha tanti stati interni quante sono le istruzioni del programma (lo stato corrente determina la istruzione in esecuzione). La PO esegue le operazioni elementari previste da ogni istruzione quindi dispone di: registri, reti combinatorie (ALU, ...), circuiti per l'instradamento dei dati (multiplexer, bus, ...).

La PO provvede all'esecuzione dei comandi del microlinguaggio (microistruzioni) tramite reti combinatorie standard e registri. La PC provvede al:

- controllo di sequenzializzazione delle microistruzioni tramite variabili di condizionamento $\{x\}$ relative allo stato interno di PO;
- invio comandi di esecuzione a PO tramite variabili di controllo $\{\alpha\} \cup \{\beta\}$;

Per quanto riguarda la realizzazione di PO e PC tramite reti sequenziali, la PO è tipicamente vista come rete sequenziale di Moore e se si utilizzano microoperazioni non parallele basta una ALU multi-funzione per realizzare le operazioni aritmetico-logiche necessarie per la elaborazione. La PC è invece molto più frequentemente realizzata secondo il modello di rete sequenziale di Mealy

tempo medio accesso:

Tempo medio di accesso alla memoria = Tasso di successo x tempo di accesso a cache +

Tasso di fallimento x Penalità di fallimento

(ns o cicli di clock)

banda passante:

Il numero medio di operazioni di I/O completabili per unità di tempo (equivalente alla quantità totale di dati trasmessi)